МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**Национальный исследовательский ядерный университет «МИФИ»**

|  |  |
| --- | --- |
|  | |
|  | **Институт  интеллектуальных кибернетических систем** |
| **Кафедра №22 «Кибернетика»** |
| Направление подготовки 09.03.04 Программная инженерия | |
| **Отчет по курсу:** | |
| **«Технологии промышленной разработки программного обеспечения. Основные понятия»** | |

Лабораторная работа 1

|  |  |  |
| --- | --- | --- |
| Группа | M24-514 | |
| Студент | Фонов Н.Д. | |  |
|  | (ФИО) | |
| Руководитель | Гагарин А.П. | |  |
|  | (ФИО) | |
|  |  | |  |
|  | |
|  | |
|  | |
|  | |
| **Москва 2024** | |

**1. Описание.**

Лабораторная работа 1 представлена в виде C# проекта (windows forms). Была выполнена доработка для обеспечения универсальности шаблона. То есть, первой строкой пользователь вносит шаблон, далее вносит данные по определенному ранее шаблону.

**2. Структура проекта.**

* lab3
  + Form1.cs
  + Program.cs
  + TemplateConstant.cs
* Прочие файлы.

В файле Form1.cs описана сама форма ввода ФИО: описан класс формы, ее методы, вся бизнес-логика.

В файле Program.cs описан класс Program, который отвечает за создание формы.

В файле TemplateConstant вынесены основные константные значения для управляющих символов.

**3. Архитектурные решения.**

Поскольку в проекте требовалось реализовать класс с реализованным функционалом перебора, то данный класс был вынесен в отдельный файл. Для реализации перебора данный класс реализует интерфейс IEnumerable. Соответственно, данный класс реализует требуемые для перебора методы.

Описание шаблона.

Шаблон обособляется символами [ и ] слева и справа соответственно. Шаблон состоит из блоков, каждый из которых описывает определенный набор символов и действий, которые должны совершаться после окончания введения данного блока.

Каждый блок состоит из 4 символов в строго определенном порядке.

Первый символ - обозначение буквы. На данный момент доступен только буквенный ввод. Варианты:

1. "w" - буква.

Второй символ - обозначение функции, которая будет модифицировать введенные буквы. Варианты:

1. "l" - приведение к нижнему регистру.

2. "u" - приведение к верхнему регистру.

3. "n" - отсутствие действий.

Третий символ - разрешенное количество букв для данного блока. Варианты:

1. "?" - возможен ввод только 1 буквы, после чего блок автоматически считается вбитым.

2. "+" - возможно произвольное кол-во букв. Для завершения блока требуется нажать соовтетвующий управляющий символ (NEXT\_TEMPLATE\_PART = пробел).

Четвертый символ - действие после завершения блока. Варианты:

1. "s" - добавить пробел после вбития блока.

2. "d" - добавить тире после вбития блока

3. "p" - добавить точку после вития блока.

4. "n" - ничего не добавлять.

Данные пользователя вносятся по шаблону. Для перехода к следующему блоку используется пробел.

Когда шаблон вбит, требуется нажать пробел, после чего данные выделятся. Далее можно вбивать следующие данные по шаблону.

**4. Тестирование и отладка.**

Для проверки корректности работы программы применялась ручная отладка.

Отладка включала в себя написание всех возможных шаблонов (в том числе и пустого шаблона) с последующим написанием данных пользователей в количестве 3 раза для каждого шаблона.

Результат ручной отладки показал, что программа работает корректно на всех заявленных выше вариантах использования шаблона.

**5. Реализация.**

Шаблон: [wu?pwu?pwu?nwl+n].

Значения: А.И.Смирнов, Б.К.Румянцев, М.А.Островский

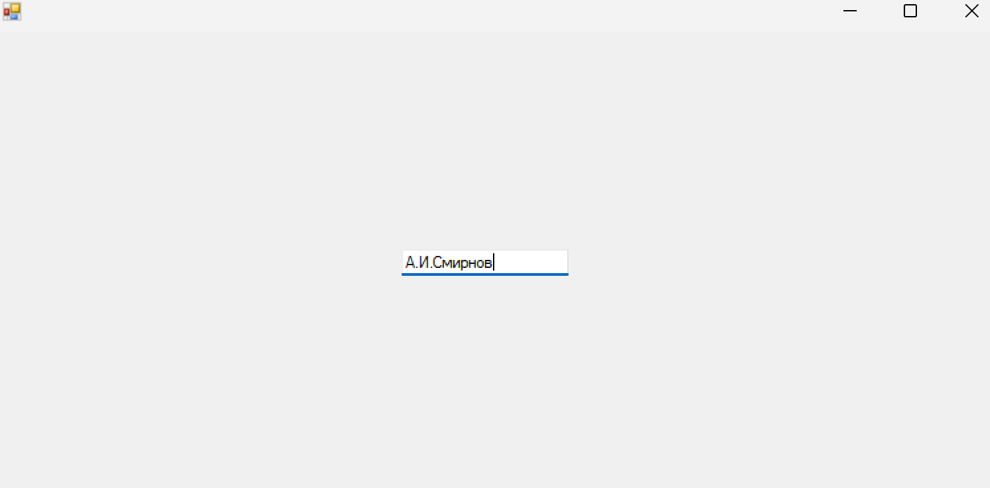


Рис. 1 – Пример 1 шаблона 1

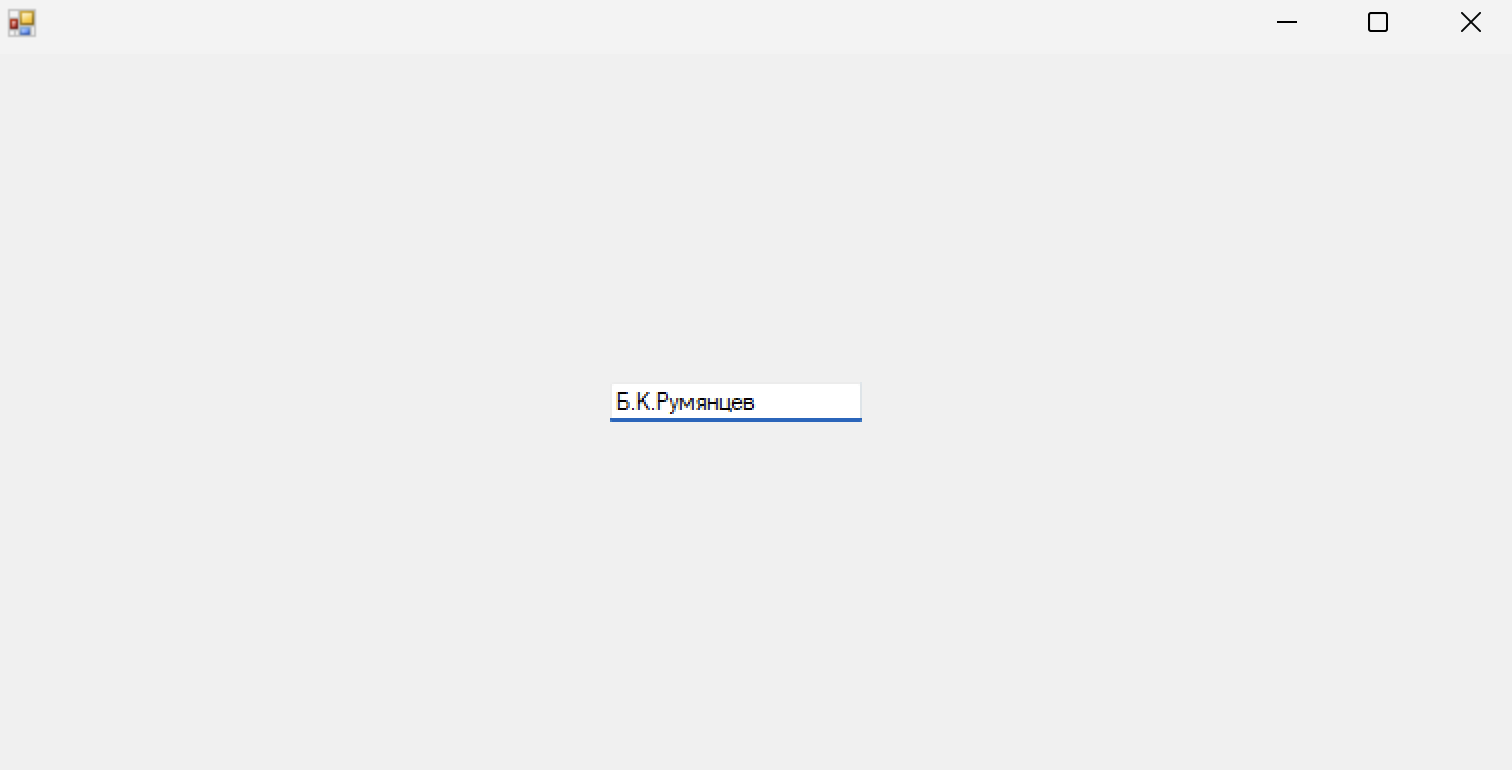


Рис. 2 – Пример 2 шаблона 1

Шаблон: [wu?pwu?nwl+dwu?nwl+n].

Значения: А.Колмогоров-Смирнов, Б.Кудряшев-Румянцев, М.Алексеев-Островский

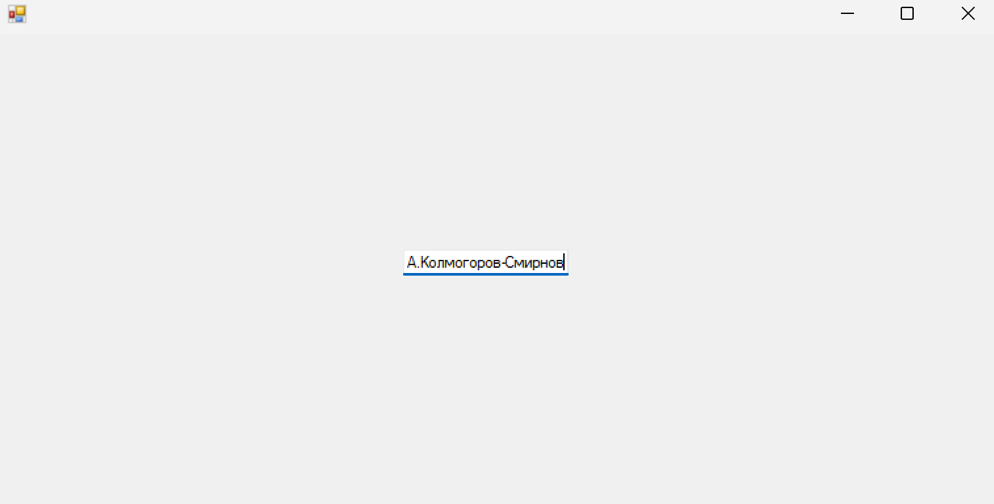


Рис. 3 – Пример 1 шаблона 2

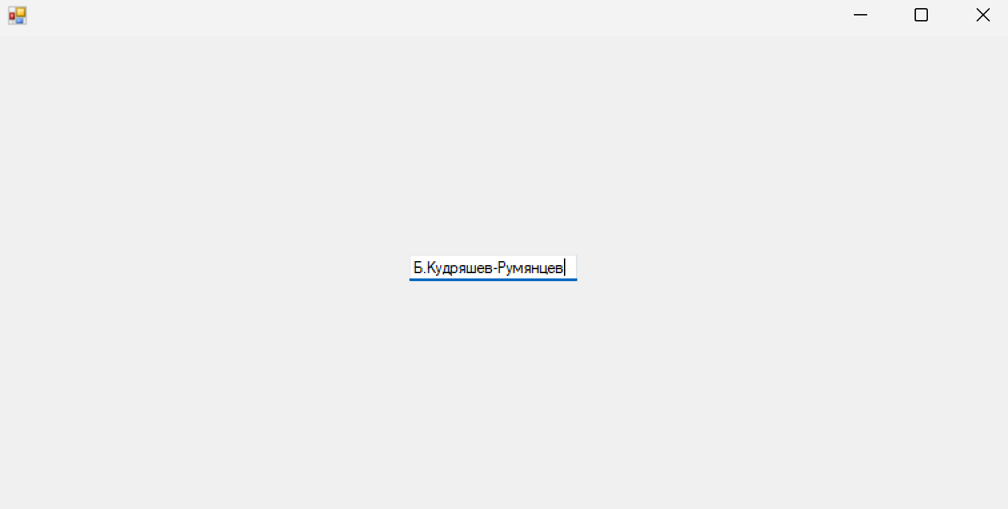


Рис. 4 – Пример 2 шаблона 2

**Приложение 1. Программный код From1.**

using lab2;

using System;

using System.Linq;

using System.Windows.Forms;

namespace lab2

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void InitializeComponent()

{

/// <summary>

/// Метод-конструтор класса формы

/// </summary>

this.textBox3 = new System.Windows.Forms.TextBox();

this.SuspendLayout();

//

// textBox3

//

this.textBox3.Location = new System.Drawing.Point(310, 164);

this.textBox3.Name = "textBox3";

this.textBox3.Size = new System.Drawing.Size(126, 31);

this.textBox3.TabIndex = 0;

this.textBox3.KeyPress += new System.Windows.Forms.KeyPressEventHandler(this.textBox3\_KeyPress\_1);

this.textBox3.KeyUp += new System.Windows.Forms.KeyEventHandler(this.textBox3\_KeyUp\_1);

//

// Form1

//

this.ClientSize = new System.Drawing.Size(764, 361);

this.Controls.Add(this.textBox3);

this.Name = "Form1";

this.ResumeLayout(false);

this.PerformLayout();

}

private string template = "";

private int templatePartIndex = 0;

bool force\_next\_stage = false;

private void processTemplateInput(KeyPressEventArgs e)

{

/// <summary>

/// Метод обработки нажатия клавиши для формирования шаблона

/// </summary>

if (textBox3.Text.Length == 0 && e.KeyChar != '[')

{

e.Handled = true;

return;

}

if (e.KeyChar == '[')

{

if (textBox3.Text.Length != 0) e.Handled = true;

return;

}

if (e.KeyChar == ']')

{

if ((textBox3.Text.Length - 1) % Constants.TEMPLATE\_PARTS\_SYMBOLS.Length != 0) e.Handled = true;

return;

}

if (textBox3.Text.Length != 0)

{

char[] allowed\_symbols = (char[])Constants.TEMPLATE\_PARTS\_SYMBOLS.GetValue((textBox3.Text.Length - 1) % Constants.TEMPLATE\_PARTS\_SYMBOLS.Length);

if (!allowed\_symbols.Contains(e.KeyChar))

{

e.Handled = true;

}

}

}

private int getWordSymbolTemplate(string template, int templatePartIndex)

{

/// <summary>

/// Метод получения символа буквы

/// </summary>

int element = Constants.TEMPLATE\_PARTS\_SYMBOLS.Length \* templatePartIndex;

return template.ElementAt(element);

}

private int getActionTemplate(string template, int templatePartIndex)

{

/// <summary>

/// Метод получения символа действия с буквой

/// </summary>

int element = Constants.TEMPLATE\_PARTS\_SYMBOLS.Length \* templatePartIndex + 1;

return template.ElementAt(element);

}

private int getAdditionalSymbolTemplate(string template, int templatePartIndex)

{

/// <summary>

/// Метод получения добавочного символа в конце части шаблона

/// </summary>

int element = Constants.TEMPLATE\_PARTS\_SYMBOLS.Length \* templatePartIndex + 3;

return template.ElementAt(element);

}

private int getRepetitionFactorTemplate(string template, int templatePartIndex)

{

/// <summary>

/// Метод получения символа повторения части шаблона

/// </summary>

int element = Constants.TEMPLATE\_PARTS\_SYMBOLS.Length \* templatePartIndex + 2;

return template.ElementAt(element);

}

private void incrimentPart()

{

templatePartIndex += 1;

}

private void transformWord(KeyPressEventArgs e)

{

/// <summary>

/// Метод изменения символа в зависимости от шаблона

/// </summary>

if (getActionTemplate(template, templatePartIndex) == Constants.TO\_LOWER) e.KeyChar = Char.ToLower(e.KeyChar);

if (getActionTemplate(template, templatePartIndex) == Constants.TO\_UPPER) e.KeyChar = Char.ToUpper(e.KeyChar);

}

private void processFioInput(KeyPressEventArgs e)

{

/// <summary>

/// обработки нажатия клавиши для формирования вывода ФИО

/// </summary>

if (e.KeyChar == Constants.NEXT\_TEMPLATE\_PART)

{

e.Handled = true;

force\_next\_stage = true;

return;

}

if (!Char.IsLetter(e.KeyChar))

{

e.Handled = true;

return;

};

transformWord(e);

}

private void textBox3\_KeyUp\_1(object sender, KeyEventArgs e)

{

/// <summary>

/// Метод обработки поднятия клавиши

/// </summary>

// обрабатываем темплейт

if (template.Length == 0)

{

if (textBox3.Text.Length != 0)

{

if (textBox3.Text.Last() == ']')

{

template = textBox3.Text;

template = template.Remove(0, 1);

template = template.Remove(template.Length - 1, 1);

textBox3.Text = null;

}

}

return;

}

// обрабатываем ввод фио

if (force\_next\_stage || getRepetitionFactorTemplate(template, templatePartIndex) == '?')

{

if (getAdditionalSymbolTemplate(template, templatePartIndex) == Constants.ADD\_DASH) textBox3.AppendText("-");

if (getAdditionalSymbolTemplate(template, templatePartIndex) == Constants.ADD\_SPACE) textBox3.AppendText(" ");

if (getAdditionalSymbolTemplate(template, templatePartIndex) == Constants.ADD\_POINT) textBox3.AppendText(".");

templatePartIndex++;

force\_next\_stage = false;

}

if (templatePartIndex >= template.Length / Constants.TEMPLATE\_PARTS\_SYMBOLS.Length)

{

templatePartIndex = 0;

textBox3.SelectionStart = 0;

textBox3.SelectionLength = textBox3.Text.Length;

return;

}

}

private void textBox3\_KeyPress\_1(object sender, KeyPressEventArgs e)

{

/// <summary>

/// Метод-триггер нажатия кнопки

/// </summary>

if (template.Length == 0)

{

processTemplateInput(e);

}

else

{

processFioInput(e);

}

}

}

}

**Приложение 2. Программный код TemplateConstant.**

using System;

using System.Windows.Forms;

public class Constants

{

// символ для перехода к следующей части шаблона

public const char NEXT\_TEMPLATE\_PART = ' ';

public const char WORD = 'w';

// начало и конец шаблона

public const char START\_TEMPLATE = '[';

public const char END\_TEMPLATE = ']';

// Повторение шаблона

public const char MULTIPLE\_SYMBOLS\_ALLOWED = '+';

public const char SINGLE\_SYMBOL\_ALLOWED = '?';

// добавления пунктуации после ввода части шаблона

public const char ADD\_SPACE = 's';

public const char ADD\_DASH = 'd';

public const char ADD\_POINT = 'p';

public const char ADD\_NONE = 'n';

// действия с буквами на поднятие клавиши

public const char TO\_LOWER = 'l';

public const char TO\_UPPER = 'u';

public const char TO\_NONE = 'n';

// массивы с управляющими символами

public static readonly char[] TEMPLATE\_FIRST\_PART\_SYMBOLS = { WORD };

public static readonly char[] TEMPLATE\_SECOND\_PART\_SYMBOLS = { TO\_LOWER, TO\_UPPER, TO\_NONE };

public static readonly char[] TEMPLATE\_THIRD\_PART\_SYMBOLS = { ADD\_DASH, ADD\_SPACE, ADD\_POINT, ADD\_NONE };

public static readonly char[] TEMPLATE\_FOURTH\_PART\_SYMBOLS = { MULTIPLE\_SYMBOLS\_ALLOWED, SINGLE\_SYMBOL\_ALLOWED };

public static readonly char[][] TEMPLATE\_PARTS\_SYMBOLS = {

TEMPLATE\_FIRST\_PART\_SYMBOLS,

TEMPLATE\_SECOND\_PART\_SYMBOLS,

TEMPLATE\_FOURTH\_PART\_SYMBOLS,

TEMPLATE\_THIRD\_PART\_SYMBOLS

};

}